Article

DOI: 10.1111/j.1468-0394.2008.00471.x

Towards ontological engineering: a process for building a domain ontology from scratch in public administration

Graciela Brusa,[1] M. Laura Caliusco[2] and Omar Chiotti[3]
(1) Dirección Provincial de Informática San Martín 2466, Santa Fe, Argentina
E-mail: gracielabrusa@santafe.gov.ar
(2) CIDISI Research Center, UTN – Facultad Regional Santa Fe, Argentina
E-mail: mcaliusc@frsf.utn.edu.ar
(3) INGAR – CONICET - UTN, Avellaneda 3657, Santa Fe, Argentina
E-mail: chiotti@santafe-ceride.gov.ar

Abstract: *The state reformation that took place in the 1990s and the technological explosion have led governments to reframe their way of working so as to be able to offer new and better services for citizens. To achieve this goal, major obstacles must be overcome, such as the problem of semantic heterogeneity that leads to more difficult recovery and integration of information from different government sectors. Although in the private sector solutions to this problem through the building of ontologies have already been set out, the characteristics of government itself have led the direct application of these practices to fail. This paper presents a process for building a domain ontology in the public sector from scratch. In addition, it presents the application of this process for building an ontology for the Budgetary Domain of Santa Fe Province (Argentina).*

Keywords: domain ontology, development process, e-government

1. Introduction

Government organizations are moving towards the use of web technologies, leading to unprecedented levels of data exchange. However, exchanging data does not mean that the data are understood. There still exists a strong need to help people and machines to understand the meaning, or semantics, of the data and applications. Nowadays, the development of ontologies tends to allow people and machines to share semantic data and software applications.

Since ontologies have gained recognition from academic and industrial areas, there are several definitions of ontology. These definitions come from different disciplines and have been used for different purposes. In information science, according to Smith (2003) ‘an ontology is a dictionary of terms formulated in a canonical syntax and with commonly accepted definitions designed to yield a lexical or taxonomical framework for knowledge representation which can be shared by different information systems communities’. In order to build an ontology, we must come to an agreement. This agreement has to follow a comprehensive ontology engineering process.

Several methodologies have been proposed to structure this process and to facilitate its development. The success of these methodologies has been demonstrated on a number of applications (Corcho *et al.*, 2005). Nevertheless, ontology development in some areas has not been as expected. One example is the public sector, which is characterized by a wide range of task and work arrangements. In addition, there are processes in which stakeholders participate because of legal rules without having the required expertise. Therefore, knowledge plays an important role in these processes (Klischewski & Lenk, 2002). Each public administration agency has its own legal rules. As a result, it is difficult but not impossible to reuse ontologies in different public administration agencies or from the private sector. Then, in public administration, the scenario will be to build an ontology from scratch involving a group of people whose knowledge and decision-making power differ significantly.

The objective of this paper is to share a process proposed for developing a domain ontology from scratch in public administration with the ontology community. With this aim, the paper is organized as follows. Section 2 discusses related work. Section 3 introduces some topics that are being used in the paper. Section 4 presents a process for building a domain ontology from scratch. Section 5 analyses the application of the process in building an ontology for the Budgetary Domain of Santa Fe Province (Argentina). Finally, Section 6 is devoted to the conclusions of the work.

2. Related work

2.1. *Technological aspects of an ontology in the public sector*

The public administration area is somehow a more promising application field for ontologies than many other e-business areas. This is because legislative knowledge is highly 'formal' by nature and, by definition, it is shared by many stakeholders (Abecker *et al.*, 2004). In the private sector, many works have been developed to improve both the technological aspects of an ontology and the methods of development. A replication of these advances in the public sector, however, is questionable due to the particular characteristics of public administration. Some examples of these particular features are as follows.

- The complex goal structure of public administration distinguishes the public sector from private business (Traunmüller & Wimmer, 2002).
- Legal norms are a standard vehicle of communication between central authorities and executive agencies. Generally, public administrations are highly regulated by legislation that is enacted on several levels (supranational, national, regional, local). Legal norms give particular meaning to administrative structures (Traunmüller & Wimmer, 2002).
- Public administration works via a complex relation of cooperation between acting entities (Traunmüller & Wimmer, 2002).
- There are four basic roles in public administration from the information system viewpoint (Apostolou *et al.*, 2005): (i) politicians who define the law; (ii) public administrators who define processes for implementing the law; (iii) programmers who implement these processes; and (iv) end-users. Since public administrators have a good knowledge about the domain, they have a key role.
- Business processes in the public sector cover a wide range of tasks and work arrangements. While some of them can be fully automated, others rely on human agency and professional knowledge and require flexibility to a large extent. Legal rules and the explicit and implicit knowledge of administrators play an important role in such processes (Klischewski & Lenk, 2002).

Thus, replicating approaches developed in the commercial domain can only be a part of the strategy that brings ontology technologies to public administration.

2.2. *Ontology development methodologies*

Ontologies have been used in different disciplines for different purposes. Several methodologies for developing ontologies have been defined (Wache *et al.*, 2001; Corcho *et al.*, 2003). Two groups of methodologies can be figured out. The first one is the group of experience-based methodologies represented by the Grüninger and Fox methodology defined in the TOVE project (1995) and by the Uschold and King methodology based on the experience of developing the Enterprise Ontology (1995). The second one is the group of methodologies that propose a set of activities to develop ontologies based on their life cycle and the prototype refinement, such as the METHONTOLOGY methodology (Gómez-Pérez *et al.*, 2004) and the Ontology Development 101 Method (Noy & McGuinness, 2001).

There is not just one correct way or methodology for developing ontologies; there are always viable alternatives. The best solution usually depends on the application that ontologists have in mind and the extensions that they anticipate. On the one hand, the ontologist could decide to use a previously defined methodology. Usually, the first group of methodologies is appropriate when purposes and requirements of the ontology are clear; the second group is useful when the environment is dynamic and difficult to understand and the objectives are not clear from the beginning (Cristani & Cuel, 2004b). On the other hand, it is common to merge different methodologies since each of them provides some different design ideas. Mainly, this merging depends on the ontology application, the tools used to develop it, and the background knowledge ontologists have.

Although several ontologies have been developed with the methodologies of the first group, they present some weaknesses: the ontology life cycle is not completely identified and these methodologies have only been tested in the business domain. With regard to the second group of methodologies, it is important to highlight that METHONTOLOGY is an approach that proposes an ontology life cycle model and it provides the most accurate descriptions of each activity. However, from the viewpoint of the public administration domain, in this methodology the cognitive representation of the reality, ontological commitments and design criteria are *implicit* in the ontology code. Furthermore, both the informal description of the ontology and the ontology implementation are often developed in separate stages. This increases the gap between real-world representations and executable systems (Cristani & Cuel, 2004a). The purpose of this paper is therefore to solve these drawbacks.

3. Preliminaries

Since ontologies have been used in different disciplines for different purposes, there are several definitions about what an ontology is and what its elements represent. The purpose of this section is to provide an introduction to some topics that are being used in the remainder of the paper.

From a pragmatic point of view, a *domain ontology* is a representational artefact whose representational units are terms, relations between them, axioms and instances. A *term* is a word or group of words representing an *entity* from the domain of discourse. A domain is a portion of the reality that forms the subject matter of a single science or technology or mode of study. The entities represented in the ontology are first order entities in reality (B. Smith *et al.*, 2006), e.g. organizations, persons, laws, procedures, among others. An *instance* is a certain individual of a corresponding entity in the domain of discourse. For example, a particular budgetary law is an instance of the entity law. A term representing an entity and its instances is related by the association *instance-of*. *Relations* represent a type of association between entities of the domain. Relations can be divided into hierarchical relations (*is-a*), bidirectional relations (e.g. *synonym* and *antonym*) and particular relations (defined by the ontology designer) (Caliusco *et al.*, 2005). *Axioms* serve to represent sentences that are always true in a domain. Finally, a *computational ontology* is a domain ontology converted into a formalized representation by using a language interpretable by a computer. The choice of this language will depend on the complexity of what one needs to represent and on the sorts of reasoning one needs to perform (B. Smith *et al.*, 2006).

A process for the development of an ontology refers to the activities that are performed to build it. An ontology can be built by using different representational artefacts depending on the potential use of the ontology, obtaining layers of representations as a result. In this way, the gap between real-world representations and executable systems is reduced. Considering these ideas, the proposed ontology building process suggests using the Unified Modelling Language (UML) to build a graphical representational artefact. The resulting artefact is a UML class diagram making the ontology more shareable. A *class* is a collection of just individuals to which a given general term is applied. The hierarchical relations between terms are represented in the UML class diagram as a *subclass* relation. Other relations are represented as associations.

In the knowledge management area, ontologies have been used for making explicit the conceptualization behind knowledge bases (O'Leary, 1998; Brewster & O'Hara, 2004). Then, in this area, ontologies have been developed from a concept orientation point of view. In contrast, the methodology proposed in this work allows us to develop an ontology as an artefact to represent entities of a given domain.

4. A process for building a public domain ontology

The process proposed in this paper is defined to satisfy the requirements of ontology development in the public administration domain. These requirements are as follows: a more adequate representation of a local domain, and an effective development of a domain ontology from scratch. Taking into account these requirements, the process is developed based on METHONTOLOGY, since it is based on the IEEE standard for software development (IEEE, 1996), and the Grüninger and Fox methodology, since it is based on the domain features of enterprise modelling and it proposes a high degree of formality. In addition, the process includes some activities of the Ontology Development 101 Method. Finally, in order to fulfil the requirements, some intermediate representations based on mature software engineering techniques are included.

The goal of this ontology development process is to build a domain ontology as a formal structure expressed in a formally defined language based on artificial intelligence techniques. Three main subprocesses compose the process: specification, concretization and implementation. The output of each subprocess is the input of the next one. In this way, integration among subprocesses is achieved, avoiding gaps between them. Figure 1 shows the relations between these subprocesses and the output of each of them. The figure is based on the software process engineering metamodel. Each subprocess consists of a set of activities that are described below.

4.1. *Subprocess 1: Specification*

The goal of the specification subprocess is to develop a domain cognitive representation and estimate the amount of effort required to devel-

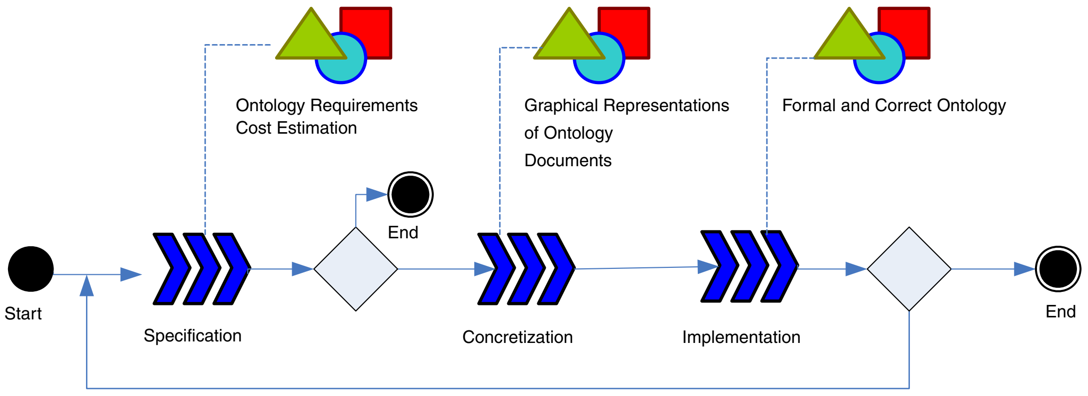


Figure 1: *Domain ontology development process.*

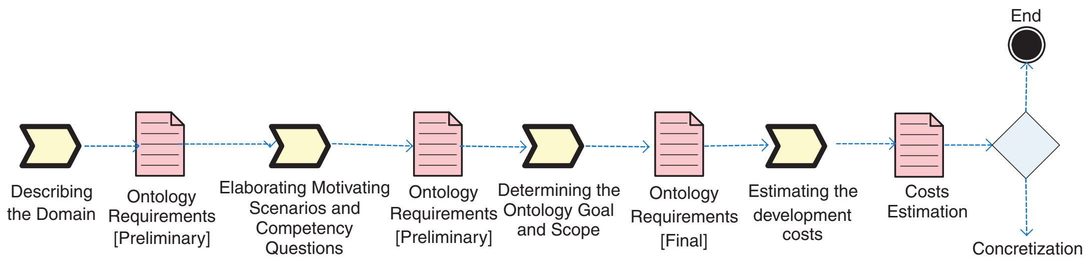

End
Describing the Domain
Ontology Requirements [Preliminary]
Elaborating Motivating Scenarios and Competency Questions
Ontology Requirements [Preliminary]
Determining the Ontology Goal and Scope
Ontology Requirements [Final]
Estimating the development costs
Costs Estimation
Concretization

Figure 2: *Specification subprocess activities.*

op a domain ontology. The input of this subprocess constitutes the need to solve semantic aspects in a certain government area and the political decision to work on it. The output is a specification document of the ontology requirements and a cost estimation document. The first document contains a domain description in natural language and the ontology goal and scope. Furthermore, this document has to include both a motivating scenario using a formal intermediate representation and competency questions. The second document contains the estimation of the costs of ontology development.

The specification subprocess consists of four activities (Figure 2): (1) describing the domain, (2) elaborating the motivating scenario and competency questions, (3) determining the ontology goal and scope and (4) estimating the development cost. These activities are described in the following.

4.1.1. Describing the domain The process suggests starting the development of an ontology by informally describing the domain of discourse. To this aim, meetings with public agents must be held. The objective of these meetings is to identify the roles played by these public agents and the domain experts and to develop a domain cognitive representation.

In public administration, the group of experts is quite heterogeneous and communication between them is ruled by legal norms. In addition, they do not have much time to attend the meetings. This group is the support to develop a cognitive representation of the domain. Then, to share it with domain experts, with the purpose of being criticized by them, it is necessary to use an appropriate representational artefact. This artefact has to be selected taking into account the background knowledge of the domain experts and the time they could spend.

4.1.2. Elaborating motivating scenarios and competency questions In order to represent the real world, Grüninger and Fox (1995) proposed deriving requirements by analysing different scenarios. Following this criterion, the second activity proposed in this process is to elaborate motivating scenarios and competency questions.

The motivating scenarios are artefacts that describe a set of requirements that an ontology should satisfy after being formally implemented in software. A motivating scenario also provides a set of intuitively possible solutions to the scenario problems. These solutions give a first idea of the informal intended semantics of the entities and relations that will be later represented in the ontology. In order to share the motivating scenarios with the people involved, it is useful to use a template as a representational artefact of them. This template could be based on those proposed to specify use cases in object-oriented methodology (Uschold & Grüninger, 1996). The process presented in this paper proposes to use the template shown in Table 1. Since this template only shows the most important information in a concise way, it is useful, when experts do not have much time, for analysing the scenarios. When a scenario is complex, it could be divided into different sub-scenarios to simplify its understanding.

After describing motivating scenarios, a set of informal competency questions based on them has to be elaborated. Informal competency

Table 1: *The scenario description template*

Scenario number	*Scenario identification*
Name	Scenario name
Description	Brief scenario description
Site	Location where the scenario occurs
Actors	People that participate in that scenario
Pre-requirements	Set of requirements that must always be met prior to the execution of the scenario
Requirements	List of needs required to execute the scenario
Normal sequence	Set of tasks that define the normal sequence of the scenario
Post-condition	Condition that must always be true just after the execution of the normal sequence
Exceptions	Actions that are not part of normal operations or standards
Legal norms	List of legal norms that are related to the scenario
Main problems	List of possible problems caused by semantic heterogeneity
Main terms	List of possible terms/concepts related to the scenario

questions are those written in natural language to be answered by the ontology once the ontology is implemented. Competency questions allow the scope of the ontology to be decided and they will serve to validate the ontology during the implementation subprocess.

4.1.3. Determining the ontology goal and scope The ontology goal represents the purpose for which the ontology will be used. An ontology could be used to assist natural-language processing, support information integration or just represent the semantics of a certain domain to support information search. Depending on the ontology goal, the entities to be represented in the ontology will be identified.

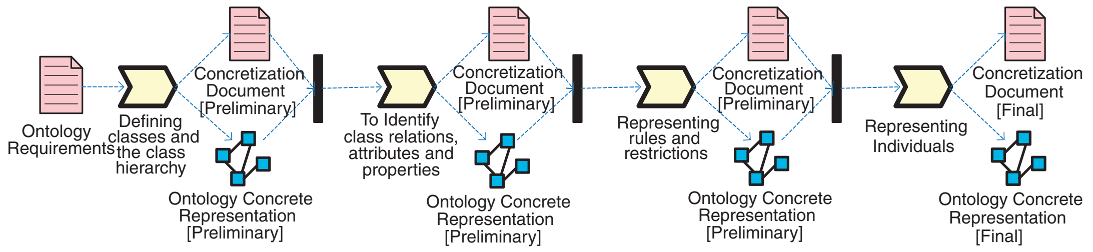
The scope limits the ontology, specifying what must be represented and what must not. It means that an entity must not be represented in the ontology if there is no competency question that uses it. Determining the ontology scope is an important activity of the specification subprocess in that it minimizes the amount of data to be analysed, especially for the extent and complexity of the public administration domain.

The definition of the ontology goal and scope was considered the first task in the 101 Method. There are two main reasons that justify defining motivating scenarios and competency questions at the beginning.

First, the ontology goal can be derived from the description of motivating scenarios. There are different goal-driven requirements analysis methods that could be applied to derive a goal of complex public administration structure from use cases. A method that adjusts to the characteristics of the public administration goal is the goal-based requirements analysis method (Anton, 1996). This method assumes that goals have not been previously and explicitly elicited from the stakeholders. To determine the goals, the analyst must work according to existing diagrams of processes or information flows, textual statements of needs, and/or additional sources of information such as transcripts of interviews with stakeholders.

Second, and according to the actors of motivating scenarios, people who will use and maintain the ontology have to be settled during this activity. In public administration, it is important to identify first the roles played by public agents. Without this previous task, it is difficult to capture the attention of public agents and to have them engaged in the ontology development process. It is necessary to have public agents engaged in the ontology use showing them the semantic problems they have and how they could solve these problems using the ontology.

4.1.4. Estimating development costs Ontology cost estimation is the activity of predicting the amount of effort required to build an ontology. This task depends on the details of the current project as regards product, personnel, and the aspects of the ontology development process. In the process presented in this paper, a sequential development process is proposed; it captures the number of participants, the expected size of the ontology, information sources used, and other relevant information that can be used in a cost



Ontology Requirements
Defining classes and the class hierarchy
Concretization Document [Preliminary]
Ontology Concrete Representation [Preliminary]
To Identify class relations, attributes and properties
Concretization Document [Preliminary]
Ontology Concrete Representation [Preliminary]
Representing rules and restrictions
Concretization Document [Preliminary]
Ontology Concrete Representation [Preliminary]
Representing Individuals
Concretization Document [Final]
Ontology Concrete Representation [Final]

Figure 3: *Concretization subprocess activities.*

estimation methodology. Taking this into consideration, ONTOCOM (Bontas *et al.*, 2006) could be used. This ontology costing artefact focuses on a sequential development process and includes mathematical equations based on research and previous project data.

In the public area this activity is very important because the project continuity depends on the approval of this document by the corresponding authorities, and sometimes the approval could involve the issue of legal norms.

4.2. Subprocess 2: Concretization

The goal of concretization is to build a graphical representational artefact that represents the ontology organizing the relevant entities identified from the domain cognitive representation developed in the previous subprocess. This artefact can be built by using a graphical modelling language. The language has to be independent of the ontology implementation languages and tools.

The development of the graphical representational artefact is the most important task in the proposed methodology. It is important to assign all the necessary time to carry out a good analysis. The domain ontology has to be agreed with domain experts. The use of a graphical representational artefact is essential to facilitate communication between ontology engineers and experts. So, software engineering techniques that could be familiar to domain experts, such as the UML (UML, 2006), can be useful.

Although UML in its standard form is not suitable for semantic representation, the use of graphical representation is suitable to enable communication between ontology engineers and domain experts (Falbo, 2004). The UML class diagram can be used to represent the ontology in term of classes and relationships between them (Cranefield & Purvis, 1999). In addition, if an ontology-based application is being constructed using object-oriented technology, it may be advantageous to use the same paradigm for representing ontologies (Cranefield, 2001). In the last few years, some meta-object facility based ontology modelling languages were defined (Caliusco *et al.*, 2005). However, there are no appropriate tools to use them yet.

Figure 3 shows a schematic representation of the concretization subprocess. The input of this subprocess is the ontology requirement specification document defined in the previous subprocess. The output is a graphical representation of the domain ontology based on UML class diagrams adding instances. The activities of this subprocess are described below.

4.2.1. Defining classes and class hierarchy The first task is to identify a list of the most relevant terms from the ontology requirement document. Initially, it is acceptable to get a comprehensive list of terms without worrying about the overlap between entities they represent, relations among the terms, or any properties that the entities may have, or whether the entities are classes or attributes (Noy & McGuinness, 2001).

The second task is to represent the terms of this list as a hierarchy of classes. With this aim, the middle-out strategy (Uschold, 1996) could be used. With this strategy, the core of basic terms has to be identified first. Then, they have to be specified and generalized if necessary.

In order to represent the class hierarchy the use of a UML class diagram is proposed, defining a class called 'Thing' as the top class of the hierarchy. If the diagram becomes complex and difficult to manage, it can be modularized by using packages. This modularization can be useful later to decide if it is necessary to build a task ontology for each package. A task ontology represents entities from a specific task. Working with different ontologies allows term reusability and usability, which are relevant goals in ontology development and differ finely. While reusability implies maximizing ontology use among different task types, usability maximizes the number of different applications using the same ontology (Jarrar, 2005).

Finally, disjoint classes, exhaustive decompositions and partitions (Horridge *et al.*, 2004) may be identified in the UML class diagram. A disjoint decomposition of a class C is a set of subclasses of C that do not have common instances and do not cover C, i.e. there can be instances of class C that are not instances of any of the classes in the decomposition. An exhaustive decomposition of a class C is a set of subclasses of C that cover C and may have common instances and subclasses, i.e. there cannot be instances of the class C that are not instances of at least one of the classes in the decomposition. Partition of a class C is a set of subclasses of C that do not share common instances and that cover C, i.e. there are no instances of C that are not instances of one of the classes in the partition.

4.2.2. Identifying class relations, attributes and properties

Once the hierarchies and their features have been identified, a table to reflect the bidirectional and user-defined relations may be elaborated, assigning names following uniform criteria and identifying domain, range, cardinality and inverse relations.

Then, the information of this table has to be represented in a UML class diagram. The UML class diagram could be used to share the developed domain cognitive representation with the experts. Furthermore, this diagram could be used later to compare it with the graphical representation of the formalized representation of the ontology. Empirical studies have demonstrated that people understand graphical notations more easily than other formalisms such as predicate logic.

4.2.3. Representing rules and restrictions

The next activity in the concretization subprocess is to represent rules and restrictions. On the one hand, in general usage a restriction is a specific type of rule that sets a finite (and generally absolute) boundary defined for a type of process or function. With regard to ontology, restriction refers to constraints imposed by the way entities are structured in reality. For example, cardinalities and allowed values express restrictions. Restrictions can be captured by the graphical representation and should not be written explicitly (Falbo, 2004). They could be represented while specifying class relations and attributes.

On the other hand, a rule is a widely accepted norm, truth, definition or qualification in the domain of discourse. In the public administration domain, rules are clearly founded in the legal norms. Then, in order to set rules, an ontological engineer has to analyse not only the scenario description templates associated with the term under consideration, but also legal norms associated with these scenarios. Rules need to be explicitly represented using a formal language, such as first order logic.

This distinction is suitable for guiding ontological engineering to represent ontology constraints.

4.2.4. Representing individuals

The last activity of the concretization subprocess is to represent individuals as instances of classes. Representing an individual in the UML diagram requires (1) choosing a class, (2) identifying the term that is represented by that class, (3) identifying the individuals associated with the term, (4) representing these individuals as instances of the class and (5) filling in the attribute values. In order to identify individuals associated with the terms represented in the ontology, it is useful to analyse the scenario description templates and competency questions.

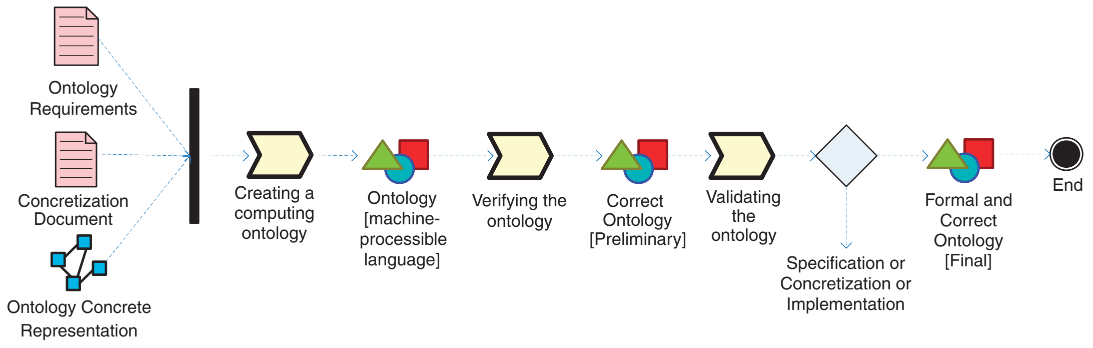

Ontology Requirements
Concretization Document
Ontology Concrete Representation
Creating a computing ontology
Ontology [machine-processible language]
Verifying the ontology
Correct Ontology [Preliminary]
Validating the ontology
Specification or Concretization or Implementation
Formal and Correct Ontology [Final]
End

Figure 4: *Implementation subprocess activities.*

4.3. Subprocess 3: Implementation

The goal of the implementation subprocess is to convert the ontology into a formalized representation interpretable by a machine, using an appropriate language with formal semantics. With this aim, the subprocess is divided into three main activities: creating a computational ontology, verifying the ontology, validating the ontology. Figure 4 shows a graphical representation of this subprocess. The activities are described in the following.

4.3.1. Creating a computational ontology Creating a computational ontology means converting the ontology into a formalized representation interpretable by a machine. There are different languages. The most relevant ones are RDF (Resource Description Framework) and OWL (Web Ontology Language) (Corcho *et al.*, 2003). The former was developed by the W3C (the World Wide Web Consortium) as a semantic network based language to describe web resources. The latter was created by a W3C Working Group called Web-Ontology (WebOnt) with the aim of making a new ontology markup language for the semantic web.

The first challenge during this task is to convert the ontology represented as a UML class diagram into one of these languages. This task is time consuming; it implies transforming composition relations into bidirectional relations considering that not all relations in UML have to be represented in the formalized ontology but only those relations that are necessary to answer competency questions.

In order to carry out this activity, an ontology development tool could be used. In the last few years, a new generation of ontology engineering environments has been developed with the aim of integrating ontology technology in new information systems. Among these environments, Protégé 3.1 (Gennari *et al.*, 2003), WebODE (Corcho *et al.*, 2002) and OntoEdit (Sure *et al.*, 2002) can be mentioned.

4.3.2. Verifying the ontology In the context of software engineering, formal verification is the act of proving or disproving the correctness of a software application as regards certain formal specifications or properties, using formal methods.

In order to prove the correctness of the ontology, consistency, completeness and conciseness have to be proved (Gómez-Pérez *et al.*, 2004).

- *Consistency*. A given representation is consistent if and only if the individual representation is consistent and no contradictory sentences can be inferred using other representations and axioms. The common errors associated with consistency are circularity errors, partition errors and semantic inconsistency errors.
- *Completeness*. In fact, neither the completeness of an ontology nor the completeness of its representations can be proved, but we can prove the incompleteness of an individual representation and thus deduce the incompleteness of an ontology. Furthermore, we can prove the incompleteness of an ontology if at least one representation is missing with regard

to the established reference framework. So, an ontology is complete if and only if

  - all that is supposed to be in the ontology is explicitly represented in it, or can be inferred;
  - each representation is complete. This is determined by figuring out what entities of the world are or are not explicitly represented, and all entities that are required but are not explicitly represented can be inferred using other representations and axioms. If it can be inferred, the representation is complete. Otherwise, it is incomplete.

    The common errors associated with completeness are incomplete class classification and partition errors (subclass partition omission and exhaustive subclass partition omission).

- *Conciseness*. An ontology is concise if it does not store any unnecessary or useless representations, if explicit redundancies do not exist between representations, and redundancies cannot be inferred using other representations and axioms. Common errors associated with redundancy are redundancies of subclass-of relations, redundancies of instance-of relations, identical formal representation of some classes, identical formal representation of some instances.

Two main types of measurement for verification can be identified: structural measures and formal measures. The first is required to represent the ontology graphically. The second implies using a reasoner.

During the ontology development process, it is acceptable to carry out a permanent and iterative verification process, taking into account that partial verifications allow propagation of identifying errors among sets of classes. The verification activity can be carried out using the support provided by ontology development tools.

4.3.3. Validating the ontology In general, validation is the process of checking if something satisfies a certain criterion. Ontology validation refers to whether the ontology really represents the real world for which it was created (Gómez-Pérez *et al*., 2004). Then, validating the ontology means checking if the ontology meets the requirements listed in the ontology requirement document.

The validation process must always be conducted against some kind of reference frame. The reference frame may be requirement specifications and competency questions.

The evaluation may be performed automatically if the competency questions are represented formally, or semi-automatically using specific heuristics or human judgement. To formalize competency questions RDF Data Query Language (RDQL) (Seaborne, 2004) and OWL-QL (Fikes *et al*., 2003) could be used. RDQL is an implementation of an SQL-like query language for RDF. It treats RDF as data and provides queries with triple patterns and constraints over a single RDF model. OWL-QL was designed for query-answering dialogues among agents using OWL. Thus, OWL-QL is suitable when it is necessary to carry out an inference in the query.

During this subprocess, communication with domain experts is essential. The ontology validation must check if competency questions are being correctly answered and if the competency questions actually pose the right questions for the ontology purpose (Falbo, 2004). In this sense, domain experts are asked to say if competency questions satisfy their needs. One simple way to calculate the ontology confidence degree is to use the formula

$$\frac{g = \sum_1^e \left[\left(\sum_1^n q \right) / n \right]}{e} \tag{1}$$

where g is the ontology confidence degree, n is the number of competency questions, q is the degree of answer given by the ontology to each competency question (satisfactory = 1; medium = 0.50; not satisfactory = 0) and e is the number of domain experts.

Verification and validation task results are reflected in a set of modifications/refinements in the specification, concretization or implementation subprocesses. The refinement process ends when the ontology confidence degree (equation (1)) is greater than a previously specified threshold, e.g. 95%.

5. A study case: an ontology for the Budgetary Domain of Santa Fe Province

The objective of this section is to present the implementation of the process described above. With this aim, the process was applied to the development of an ontology for the Budgetary Domain of Santa Fe Province (Argentina). Since then, this approach has been used to create a Government Budgetary Ontology (Brusa *et al.*, 2006a, 2006b).

5.1. Specification subprocess

5.1.1. Activity 1: Describing the domain In this activity, the information system that supports the provincial budget formulation and its related documentations was studied and revised. Furthermore, meetings with a group of experts were held. This group was constituted by public officials responsible for the whole budget formulation process in the Executive Power, expert professionals of the Budget Committee of the Legislative Power, public agents of the administrative area in charge of creating their own budget and software engineers who provided informatics support for these tasks.

In the following, a brief description of the domain is presented.

Budgetary and financial domain The budget of a government is a plan for the intended revenues and expenditures of that government. The budget is prepared by different entities in different government areas. Specifically, in Santa Fe Province (Argentina) these entities are

- Executive Power: this government entity elaborates the Provincial Budget Draft and is formed by a ruling organism and executing organisms. The first defines all activities for formulating a budget and the others execute these activities.
- Legislative Power: this government entity passes the Annual Budget Law.

Along the life cycle of a budget, the evaluation and control of current financial resources are carried out, and all of them are assigned to goods and services production. Table 2 shows these steps in detail.

Table 2: *Budget life cycle steps*

1.	Initiate fiscal year and distribute classifiers
2.	Prepare preliminary budget and resources estimation
3.	Define budgetary policy and expenses projection
4.	Determine expenses top
5.	Formulate budget project draft
6.	Present budget project draft to legislature
7.	Approve budget in legislature
8.	Elaborate a new budget based on budget law
9.	Distribute budget for execution
10.	Elaborate budgetary modifications
11.	Program budget executing
12.	Reconduct budget
13.	Closure fiscal year

There is common information for all budget life cycle stages. A classifier is an information classification. In the particular case of budgetary classifiers, they show different ways of classifying information. Budgetary classifiers are derived from the expenses and resources transactions of public institutions. This level of detail is necessary to express, formally and in a precise way, revenue sources (resource classifiers) and expense characteristics (expense classifiers). This involves specifying goods and services to acquire in a government programme, their geographical location and destination. Classifiers are used for programming, analysing and controlling the economic and financial management of public institutions. Classifiers used in this work are institutional classifier, expense object, geographical location, finality function, resource item, financing source and programmatic category.

Two situations were analysed in this work where the availability of semantic information associated with budgetary data is critical: budget formulation and approval tasks. In the first case, only government staff with specific knowledge can be involved, concentrating a great responsibility on a few people. In the second case, semantic information is necessary for analysing budgetary data and then having the budget law passed. Here, this is more complex because all legislators must vote and most of them have no specific knowledge. For simplicity, only the formulation stage for the expense budget was considered for this case study.

5.1.2. Activity 2: Elaborating motivating scenarios and competency questions A motivating scenario describing the main features of the local budget formulation was defined. In order to arrive at a precise description of the scenario, several meetings with domain experts were held. Since the main scenario was very complex, it was defined including different sub-scenarios that describe in detail one step of the normal sequence. The inclusion is expressed with the keyword *include* followed by the name of the scenario that is included. The template that represents the main scenario of local budget formulation is shown in Table 3.

Based on this template scenario description simple and complex competency questions were derived. As an example, some of them are shown in Table 4.

5.1.3. Activity 3: Determining the ontology goal and scope The ontology goal is to represent the set of entities involved in the budget formulation task to be mainly used by public agents when they have to define a budget for the next fiscal year.

This ontology only considers the needs for creating an analytic budget with entities related to expenses. It does not consider the entities related to other stages such as budgetary executing, accounting, payments, purchases or fiscal year closure. Therefore, it includes general enti-

Table 3: *Scenario of local budget formulation*

Scenario number	1
Name	Local Budget Formulation
Description	Necessary tasks to estimate expenses for the following year, which will be integrated with the other government jurisdictions for outlining the Draft Local Budget
Site	Executing organism of a jurisdiction
Actors	• Public agents uncharged jurisdictional budget • Ruling organism agents • Public agents from areas of a jurisdiction
Pre-requirements	• Budgetary policy defined • Expenses classifiers received from ruling organism • Reference documentation
Associated requirements	• Trained agents in budget formulation tasks • Advisory agents from ruling organism
Normal sequence	STEP ACTION 1 To receive expenses estimations from jurisdiction areas 2 To bring support to this area for elaborating own expenses programs 3 To integrate all expenses programs for jurisdiction 4 To create programming categories; this includes programming categories formulation 5 To create the jurisdictional budget project; this includes jurisdictional budget project formulation 6 To load budget in informatics system and send it to ruling organism 7 To receive approved jurisdictional budget from ruling organism
Post-condition	• Jurisdictional expense budget project • Jurisdictional programmatic categories
Exceptions	STEP ACTION 5 To consult the rector organism on the different aspects of formulating the budget 7 To modify the budget if it is not approved
Main problems	• Too much time spent in clarifying conceptual doubts • Major problems when an agent must be replaced in key work places • The whole process is highly dependent on a few people's knowledge
Main terms	Budgetary classifier, expenses classifier, institutional, programmatic category, geographic, expenses object, financing source and finality function classifiers, . . .

ties for the budget life cycle and specific entities for the formulation stage.

5.2. Concretization subprocess

5.2.1. Activity 1: Defining classes and class hierarchy In this task, a list of terms that represent the most important entities in the budget formulation domain was created using the middle-out strategy. The list is shown in Table 5. It does not include partial or total overlapping of terms, synonyms, properties or relations.

In order to identify classes, those terms with independent existence from the key terms list were identified. A UML diagram (UML, 2006) was elaborated with the hierarchy relations among these terms.

This UML representation was useful to verify the ontology scope and to discover two granularity levels for budgetary domain entities. Then, it was necessary to make an important design decision: working with two ontologies. One of them is the domain ontology, shown in Figure 5, which contains the general terms for representing the budget life cycle and a coarse granularity is adequate. The other, the formulation ontology, contains the specific terms for formulating a budget. This is a task ontology (Gómez-Pérez *et al*., 2004) since it represents entities related to a specific task and a fine granularity is required. So, the list of key terms and hierarchical relations was modified and competency questions were grouped depending on the ontology terms they were related to.

From the class hierarchy diagram, disjoint classes, exhaustive decompositions and partitions were identified. As an example (Figure 5), the finality function, financing source, expense object, programmatic category, geographical location and institutional classifier can be mentioned as disjoints. Furthermore (Figure 5), the classes Expenses Classifier and Resource Classifier make up an exhaustive decomposition of the class Budgetary Classifier. This is because there is no classifier that is not an instance of at least one of those classes, and the classes can have common instances. Finally, there are no partitions in this scenario. It is always convenient to begin with primitive classes, analysing which of them are disjoint and verifying if that condition does not produce instance absences.

Table 4: *Competency questions*

Simple questions	*Complex questions*
Which are the budget states?	Which are the Department and Province for the District 'Santa Fe'?
Which are the budgetary classifiers?	Which are the sector and subsector for the Main Administration?
Which are the expenses classifiers?	
Which are the resource classifiers?	

5.2.2. Activity 2: Identifying class relations, attributes and properties Once the hierarchies and their features have been identified a table may be created to reflect the relations, assigning names according to a uniform criterion, and identifying domain and range, cardinality and inverse relations. An example is shown in Table 6. The relation direction depends on competency questions to be solved and the possible conflicts with other represented class restrictions.

Table 5: *Key term list*

Activity	*Budgetary fiscal year*	*Financial administration*	*Program*
Budget	Budgetary policy	Financing source	Subprogram
Budget analytic	Budgetary top	Geographical location	Program executer unit (UEP)
Budget approved	Executing organism	Institutional classifier	Programmatic category project
Budget project draft	Expenses	Institution	Project
Budget synthetic	Expenses classifier	Financial administrative service (SAF)	Public funds administrative service (SAFOP)
Budget states	Expense object	Jurisdiction	Ruling organism
Budgetary classifier	Finality function	Jurisdiction government	Resource

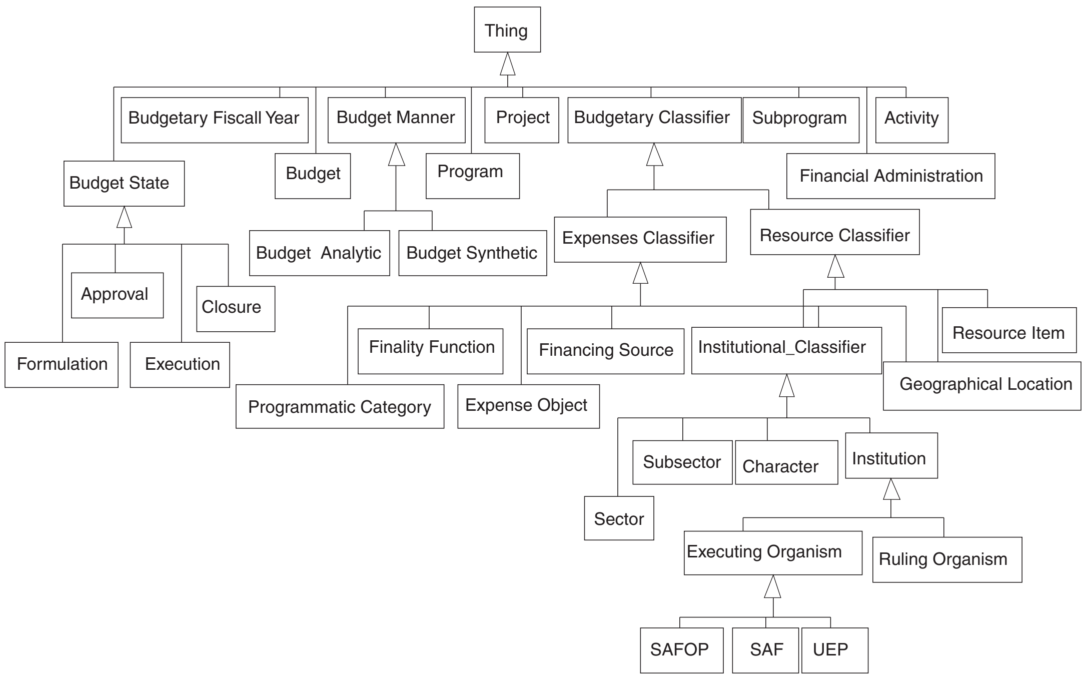


Figure 5: *The class hierarchy in UML.*

Table 6: *An excerpt of the relation table of the budgetary ontology*

Class name	*Relation*	*Cardinality*	*Class name*	*Inverse relation*
Institutional classifier	inst-include-sec	1	Sector	sec-isPartOf-Inst
Institutional classifier	inst-include-sbsec	1	Subsector	sbsec-isPartOf-Inst
Institutional classifier	inst-include-char	1	Character	char-isPartOf-Inst
Sector	sec-isPartOf-Inst	1, *n*	Institutional classifier	inst-include-sec
Subsector	sbsec-isPartOf-Inst	1, *n*	Institutional classifier	inst-include-sbsec
Character	char-isPartOf-Inst	1, *n*	Institutional classifier	inst-include-char
Institutional classifier	ins-has-SAF	1	SAF	SAF-correspond-inst

The last step of this task is to capture in the graphic representational artefact the relations and restrictions listed in the relation table. An excerpt of these relations is shown in Figure 6.

5.2.3. Activity 3: Representing rules and restrictions The first step of this task is to analyse the restrictions represented in the last task and to add those that have not been represented. Then, taking into account the scenario description templates and legal norms, several rules were identified and represented in natural language. After that, these rules were formally represented in description logic. For example, the following axiom states that a programmatic category is an expenses classifier and, if it has a subprogram, it has to be associated to a program.

$$\text{ProgrammaticCategory} \subseteq \text{ExpensesClassifier} \wedge \forall \text{hasSubprogram}.(\exists \text{hasProgram}.\text{Program})$$

Finally, the axioms have to be analysed both individually and in groups of classes to verify whether closure restrictions are required. Closure restrictions are a statement of universal restriction which means that a property can only be satisfied by an exhaustive list of conditions.

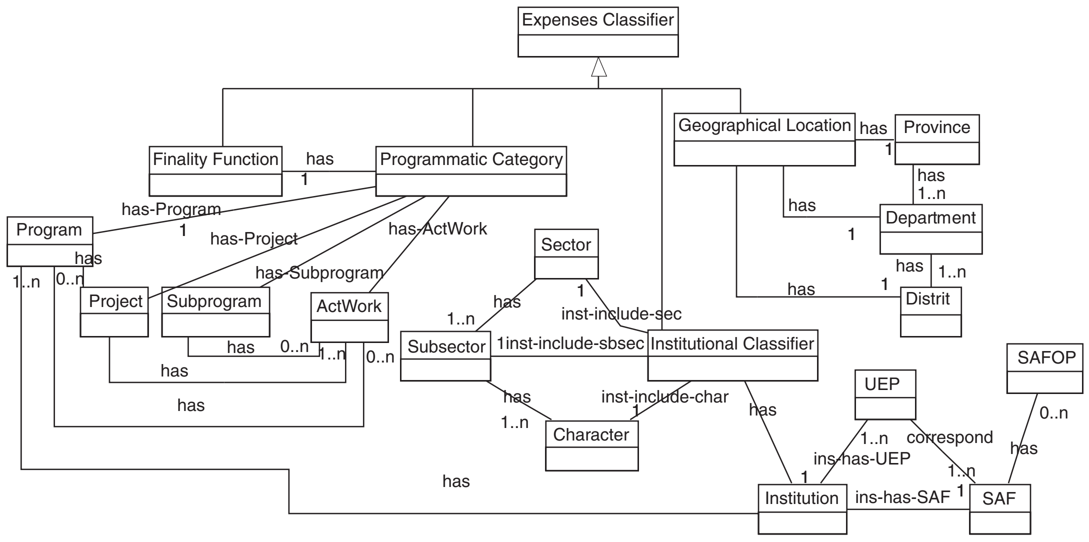


Figure 6: *An excerpt of the UML class diagram.*

Table 7: *An excerpt of the instance table of the budgetary ontology*

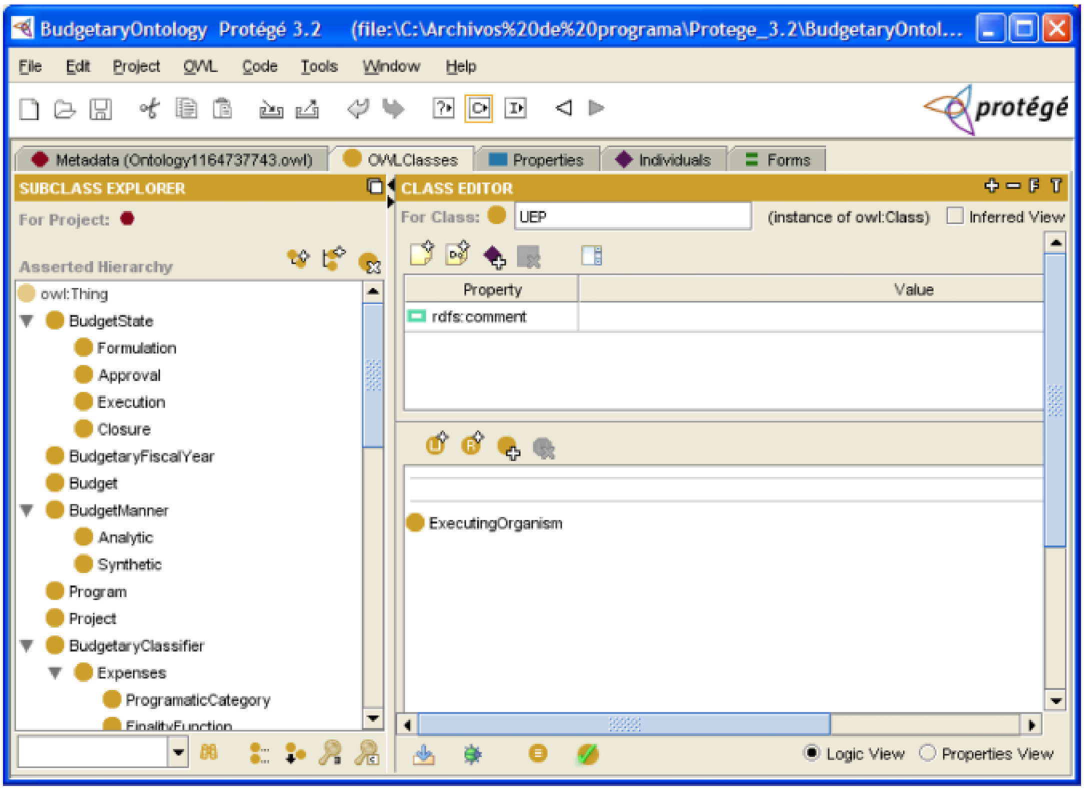
Class name	*Instance name*	*Relation*	*Value*
Institutional classifier	Institutional_111	cod-institutional	1.1.1
		inst-include-sec	1 – No financial local public sector
		inst-include-sbsec	1 – Local administration
		inst-include-char	1 – Main administration
	Institutional_212	cod-institutional	2.1.2
		inst-include-sec	2 – Financial local public sector
		inst-include-sbsec	1 – Official banking system
		inst-include-char	2 – Official banks

5.2.4. Activity 4: Representing individuals The final activity for arriving at an ontology is to represent individuals as instances in the UML class diagram. First, an instance table has to be built analysing the individuals of the domain. Each instance should be provided with its name, the name of the term it is associated with, and its attribute values, if known, as shown in Table 7. Then, each instance has to be associated with the corresponding class represented in the UML class diagram.

5.3. The implementation subprocess

5.3.1. Activity 1: Creating a computational ontology In order to implement the ontology, we chose Protégé because it is extensible and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development (Knublauch *et al.*, 2005). Protégé ontologies can be exported into different formats including RDF Schema (RDFS) (Brickley & Guha, 2004) and OWL (M. Smith *et al.*, 2004). In particular, the budgetary ontology was implemented in OWL. Figure 7 shows the logical view window of Protégé.

5.3.2. Activity 2: Verifying the ontology During the verification activity, experiences of CO-ODE Project (Knublauch *et al.*, 2005) and practical experience of teaching OWL-DL reported by Rector *et al.* (2004) were taken into account. The verification phase was incremental and continuous to avoid future propagation of errors. It was divided into a formal verification and graphical verification.


BudgetaryOntology Protégé 3.2 (file:\C:\Archivos%20de%20programa\Protege_3.2\BudgetaryOntol...
File Edit Project OWL Code Tools Window Help
protégé
Metadata (Ontology1164737743.owl)
OWLClasses
Properties
Individuals
Forms
SUBCLASS EXPLORER
For Project:
Asserted Hierarchy
CLASS EDITOR
For Class:
Logic View
Properties View

owl:Thing
BudgetState
Formulation
Approval
Execution
Closure
BudgetaryFiscalYear
Budget
BudgetManner
Analytic
Synthetic
Program
Project
BudgetaryClassifier
Expenses
ProgramaticCategory
FinalityFunction

For Class: UEP (instance of owl:Class) Inferred View

Property	Value
rdfs:comment	

ExecutingOrganism

Figure 7: *Protégé logical view window.*

On the one hand, to conduct a formal verification, consistency validation and classification from Protégé and Racer reasoner (Haarslev & Möller, 2001) were used. When a class is unsatisfiable, Protégé highlights it. There are different causes of dissatisfaction which could generate a propagation of errors. At this point, it is very important how classes are defined (disjoint, isSubclassOf, Partial Class, Defined Class etc.) and their restrictions (unionOf, allValuesFrom etc.). The classification process can be invoked either for the whole ontology or for selected subtrees only. When the test is done on the whole ontology, errors are searched beginning with minor level classes in the hierarchy for minimizing error propagation.

On the other hand, to graphically verify the ontology implementation with its concretization (the UML class diagram representation), graphics using the OWLViz and Ontoviz plug-ins were generated and compared with UML diagrams. OWLViz allows class hierarchies to be viewed in OWL ontology (Figure 8). Then, this ontology can be compared with the UML class hierarchy (Figure 5).

OntoViz generates diverse combinations of graphics with all relations represented in the ontology, instances and attributes. As an example, Figure 9 shows the main relations of the class Institutional Classifier with other classes, and one instance of this class that represents the way in which the Local Administration is classified. OntoViz allows several disconnected graphs to be visualized at once. These graphs are suitable for comparing the ontology concretization sketched in a UML class diagram (Figure 6) and in the instance table (Table 7) with its implementation.

5.3.3. Activity 3: Validating the ontology Ontology validation refers to whether the ontology really represents the actual world for which it was created. The goal is to prove that the real world is compliant with that world formally represented.

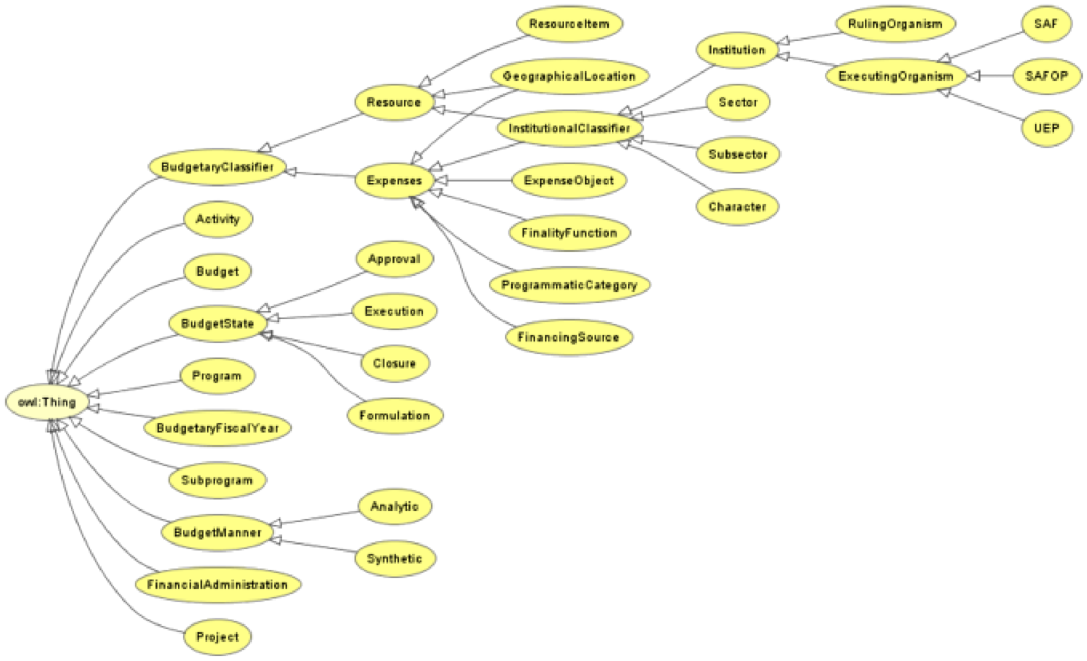
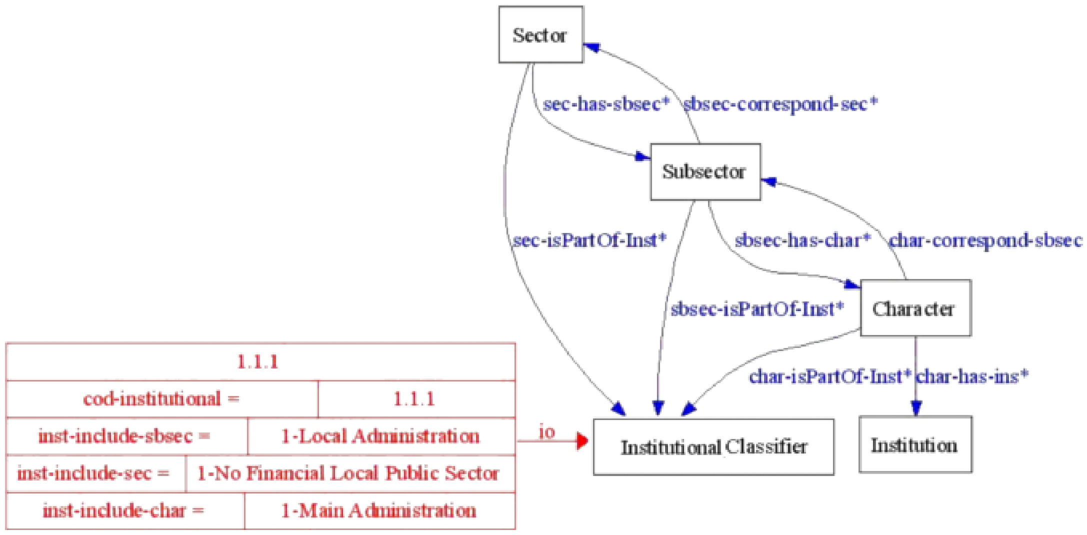


Figure 8: *Budgetary ontology taxonomy in OWLViz.*



1.1.1	
cod-institutional =	1.1.1
inst-include-sbsec =	1-Local Administration
inst-include-sec =	1-No Financial Local Public Sector
inst-include-char =	1-Main Administration

Figure 9: *Main relations and an instance of Institutional Classifier class in OntoViz.*

In order to validate the ontology, it is necessary to verify whether the ontology correctly answers the competency questions. With this aim, the competency questions were codified using RDQL and then the results were checked by the domain experts. A possible RDQL query is

```
SELECT ?x ?y ?z ?nsec ?nsbsec
WHERE (?x, <adm:sec-has-sbsec>,?y)
      (?y, <adm:bsec-has-char>,?z)
      (?z, <rdfn:label>,'1-Main
      Administration')
      (?x, <rdfn:label>, ?nsec),
      (?y, <rdfn:label>, ?nsbsec)
USING rdfn FOR http://www.w3.org/2000/01/
      rdf-schema#
      adm FOR http://protege.stanford.edu/
```

This RDQL query codifies the competency question 'Which is the sector and subsector for the Main Administration?'. The query contains three main clauses: SELECT, WHERE and USING. In the SELECT clause we have defined five variables from which a valid RDF triple could be created in the WHERE clause. In the first line of the WHERE clause a set of (sector, sec-has-sbsec and subsector) triple is retrieved. Then, in the second line, the character of each subsector is obtained and in the third line has been restricted to only that sector and subsector whose character is '1 – Main Administration'. Finally, in the fourth and fifth lines, the labels of this sector and subsector are retrieved. Both, the *rdfn* and the *adm* prefixes are defined in the USING clause, representing the commonly used prefix for RDF and the URI for the schema of the ontology respectively.

Finally, the queries were implemented using Jena Toolkit (McBride, 2002). Jena is a Java framework that provides an API for creating and manipulating RDF models.

6. Conclusions

Successful implementation and application of ontologies in public administration depends on the availability of appropriate methodologies. In this paper, a process for building a domain ontology from scratch in public administration is proposed. This process aims to meet the requirements of an ontology development in this area. The requirements are an explicit representation of a local domain, and an effective development of a domain ontology from scratch involving different stakeholders. Furthermore, the process proposes to represent the cognitive representation by using a graphical language following some ontology implementation characteristics, trying to fill the gap between domain cognitive representation and its implementation.

In addition, a case study has been presented. This case study is based on the ontology development for the Budgetary Domain of Santa Fe Province (Argentina).

Building domain ontologies is not a simple task when domain experts have no background knowledge of engineering techniques and/or they have not much time to spend during the task of supporting the domain cognitive representation. Then, the use of graphical representation is crucial in order to facilitate communication between ontology engineers and domain experts. The process discussed in this paper proposes the use of the UML class diagram as an intermediate ontology representation artefact.

References

Abecker, A., N. Stojanovic and R. Studer (2004) An approach for the change management in the e-government domain, in *Proceedings of the 2nd International Conference on Knowledge Economy and Development of Science and Technology*, Beijing: Institute of Computer Technology, Chinese Academy of Sciences, and Institute of Knowledge Economy (Tokyo). 1080–1097.

Anton, A.I. (1996) Goal identification and refinement in the specification of software-based information systems, PhD Thesis, Georgia Institute of Technology, Atlanta, GA.

Apostolou, D., L. Stojanovic, T. Lobo and B. Thoenssen (2005) Towards a semantically-driven software engineering environment for e-government, in *Proceedings of TCGOV 2005*, M. Böhlen, J. Gamper, W. Polasek and M.A. Wimmer (eds), LNAI 3416, Berlin: Springer, 157–168.

Bontas, E., C. Tempich and Y. Sure (2006) ONTOCOM: a cost estimation model for ontology engineering, in *Proceedings of the 5th International Semantic Web Conference*, LNCS 4273, Berlin: Springer, 625–639.

Brewster, C. and K. O'Hara (2004) Knowledge representation with ontologies: the present and future, *IEEE Intelligent Systems*, **19** (1), 72–81.

Brickley, D. and R.V. Guha (2004) *RDF Vocabulary Description Language 1.0: RDF Schema*, W3C Recommendation, at http://www.w3.org/TR/rdf-schema/.

Brusa, G., M.L. Caliusco and O. Chiotti (2006a) Building ontology in public administration: a case study, in *Proceedings of the 1st International Workshop on Applications and Business Aspects of the*

Semantic Web at the 5th International Semantic Web Conference, E. Pasalaru, B. Simperl, M. Hepp and C. Tempich (eds), 16–30.

BRUSA, G., M.L. CALIUSCO and O. CHIOTTI (2006b) A process for building a domain ontology: an experience in developing a government budgetary ontology, in *Conferences in Research and Practice in Information Technology*, M.A. Orgun and T. Meyer (eds), Hobart: Australian Computer Society, Vol. 72, pp. 7–15.

CALIUSCO, M.L., M.R. GALLI and O. CHIOTTI (2005) Contextual ontologies for the semantic web – an enabling technology, in *Proceedings of the 3rd Latin American Web Congress*, New York: IEEE Computer Society, 98–101.

CORCHO, O., M. FERNÁNDEZ-LÓPEZ, A. GÓMEZ-PÉREZ and O. VICENTE (2002) WebODE: an integrated workbench for ontology representation, reasoning, and exchange, in *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW '02)*, A Gómez-Pérez and V.R. Benjamins. (eds), LNAI 2473, Berlin: Springer, 138–153.

CORCHO, O., M. FERNÁNDEZ-LÓPEZ and A. GÓMEZ-PÉREZ (2003) Methodologies, tools and languages for building ontologies. Where is the meeting point?, *Data and Knowledge Engineering*, **46**, 41–64.

CORCHO, O., M. FERNÁNDEZ-LÓPEZ, A. GÓMEZ-PÉREZ and A. LÓPEZ-CIMA (2005) Building legal ontologies with METHONTOLOGY and WebODE, in *Law and the Semantic Web. Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications*, LNCS 3369, Berlin: Springer, 142–157.

CRANEFIELD, S. (2001) UML and the semantic web, in *Proceedings of the 1st Semantic Web Working Symposium*, Stanford, CA, 113–130.

CRANEFIELD, S. and M. PURVIS (1999) UML as an ontology modelling language, in *IJCAI '99 Workshop on Intelligent Information Integration*, D. Fensel, C. Knoblock, N. Kushmerick and M. C. Rousset (eds), CEUR Workshop Proceedings **23**, 5.1–5.8 (http://CEUR-WS.org/Vol-23/).

CRISTANI, M. and R. CUEL (2004a) A comprehensive guideline for building a domain ontology from scratch, in *International Conference on Knowledge Management (I-KNOW '04)*, Know Center Graz, 205–212.

CRISTANI, M. and R. CUEL (2004b) Methodologies for the semantic web: state-of-the-art of ontology methodology, *SIGSEMIS Bulletin*, Theme: SW Challenges for KM, **1** (2), 103–136.

FALBO, R.A. (2004) Experiences in using a method for building domain ontologies, in *Proceedings of the 16th International Conference on Software Engineering and Knowledge Engineering, International Workshop on Ontology in Action*, Banff, 474–477.

FIKES, R., P. HAYES and I. HORROCKS (2003) OWL-QL – a language for deductive query answering on the semantic web, *Report*, KL Laboratory, Stanford University

GENNARI, J., M.A. MUSEN, R.W. FERGERSON, W.E. GROSSO, M. CRUBEZY, H. ERIKSSON, N.F. NOY and S.W. TU (2003) The evolution of Protégé: an environment for knowledge-based systems development, *International Journal of Human–Computer Studies*, **58** (1), 89–123.

GÓMEZ-PÉREZ, A., M. FERNÁNDEZ LÓPEZ and O. CORCHO (2004) *Ontological Engineering with Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*, London: Springer.

GRÜNINGER, M. and M.S. FOX (1995) Methodology for the design and evaluation of ontologies, in *IJCAI Workshop on Basic Ontological in Knowledge Sharing*, D. Skuce (ed.), 6.1–6.10.

HAARSLEV, V. and R. MÖLLER (2001) Description of the RACER system and its applications, in *International Workshop on Description Logics (DL '01)*, C.A. Goble, D.L. McGuinness, R Möller and P.F. Patel-Schneider (eds), CEUR Workshop Proceedings **49**, 132–141 (http://CEUR-WS.org/Vol-49/).

HORRIDGE, M., H. KNUBLAUCH, A. RECTOR, R. STEVENS and C. WROE (2004) *A Practical Guide to Building OWL Ontologies using the Protégé-OWL Plugin and CO-ODE Tools Edition 1.0*, University of Manchester.

IEEE (1996) *IEEE Standard for Development Software Life Cycle Processes*, New York: IEEE Computer Society, IEEE Std 1074-1995.

JARRAR, M. (2005) Towards methodological principles for ontology engineering, PhD Thesis, Vrije Universiteit Brusell.

KLISCHEWSKI, R. and K. LENK (2002) Understanding and modelling flexibility in administrative processes, in *Proceedings of EGOV 2002*, R. Traunmuller and K. Lenk (eds), LNCS 2456, Berlin: Springer, 129–136.

KNUBLAUCH, H., M. HORRIDGE, M. MUSEN, A. RECTOR, R. STEVENS, N. DRUMMOND, P. LORD, N. NOY, J. SEIDENBERG and H. WANG (2005) The Protégé OWL experience, in *Workshop on OWL: Experiences and Directions, 4th International Semantic Web Conference*, B. Cuenca Grau, I. Horrocks, B. Parasia and P. Patel-Schneider (eds), CEUR Workshop Proceedings 188 (http://CEUR-WS.org/Vol-188/).

MCBRIDE, B. (2002) Jena: A semantic web toolkit, *IEEE Internet Computing*, **6** (6), 55–59.

NOY, N. and D. MCGUINNESS (2001) Ontology development 101: A guide to creating your first ontology, *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*.

O'Leary, D. (1998) Using AI in knowledge management: knowledge bases and ontologies, *IEEE Intelligent Systems*, **13** (3), 34–39.

Rector, A., N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang and C Wroe (2004) OWL pizzas: practical experience of teaching OWL-DL: common errors and common patterns, in *Proceedings of the European Conference on Knowledge Acquisition*, E. Motta and N. Shadbolt (eds), LNAI 3257, Berlin: Springer, 63–81.

Seaborne, A. (2004) RDQL – a query language for RDF, W3C Submission, at http://www.w3.org/Submission/2004/.

Smith, B. (2003) Ontology, in *Blackwell Guide to the Philosophy of Computing and Information*, L. Floridi (ed.), Oxford: Blackwell, 155–166.

Smith, B., W. Kusnierczyk, D. Schober and W. Ceusters (2006) Towards a reference terminology for ontology research and development in the biomedical domain, in *Proceedings of KR-MED 2006, Biomedical Ontology in Action International Workshop*, S. Schulz, B. Smith and F. Neuhaus (eds), Baltimore, MD.

Smith, M., C. Welty and D. McGuinness (2004) *OWL Web Ontology Language Guide*, W3C Recommendation 10, at http://www.w3.org/TR/owl-guide/.

Sure, Y., M. Erdmann, J. Angele, S. Staab, R. Studer and D. Wenke (2002) OntoEdit: collaborative ontology engineering for the semantic web, in *Proceedings of the 1st International Semantic Web Conference*, LNCS 2342, Berlin: Springer, 221–235.

Traunmüller, R. and M. Wimmer (2002) KM for public administration: focusing on KMS feature requirement, in *Practical Aspects of Knowledge Management*, LNAI 2569, Berlin: Springer, 314–325.

UML (2006) *Unified Modeling Language*, at http://www.uml.org/.

Uschold, M. (1996) Building ontologies: towards a unified methodology, in *16th Annual Conference of the British Computer Society Specialists Group on Expert Systems*, I. Watson (ed.), Cambridge.

Uschold, M. and M. Grüninger (1996) Ontologies: principles, methods and applications, *Knowledge Engineering Review*, **11** (2), 93–155.

Uschold, M and M. King (1995) Towards a methodology for building ontologies, in *IJCAL'95 Workshop on Basic Ontological Issues in Knowledge Sharing*, D. Skuce (ed.), 6.1–6.

Wache, H., T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hübner (2001) Ontology-based integration of information – a survey of existing approaches, in *Proceedings of the IJCAI-01 Workshop: Ontologies and Information Sharing*, 108–117.

The authors

Graciela Brusa

Graciela Brusa received her Bachelors degree in applied mathematics in 1980 from UNL – Litoral National University, Santa Fe, Argentina, and her Masters in information systems engineering from Universidad Technológica Nacional – Facultad Regional Santa Fe (UTN-FRSF), Argentina, in 2006. She is working in the public sector, in particular in the informatics and communications areas. Her principal interests are such themes as knowledge management and ontologies.

M. Laura Caliusco

M. Laura Caliusco received her degree in information systems engineering in 1999 and her PhD in information systems from UTN-FRSF, Santa Fe, Argentina, in 2005. She is now an Associate Professor at FRSF. She was elected post-doctoral fellow of Argentina's National Council of Scientific Research (CONICET) in 2005. At present, she is working as a research scientist at the I + D Center in Information Systems Engineering (CIDISI). Her interests are primarily in the areas of software agents, knowledge discovery, e-collaboration, context, and ontology.

Omar Chiotti

Omar Chiotti received his degree in chemical engineering in 1984 from UTN – Facultad Regional Villa María, Córdoba, Argentina, and his PhD in chemical engineering from Universidad Nacional del Litoral, Santa Fe, Argentina, in 1989. Since 1984, he has been working for CONICET as a researcher. He has been a full Professor of Information Systems Engineering at UTN-FRSF since 1986. At present, he is the director of the I + D Center in Information Systems Engineering (CIDISI) and the director of the PhD in engineering. His current research interests include decision support systems, data warehouse, e-collaboration, and multi-agent systems.